Advanced Data Science & Python for Stock Analysis

Final Project

Part 3 - Trading Strategy and Backtesting

Creator: Wendy (Aobo) Liu

Packages Installation and Setup

In [4]:
```
!pip install configparser
!pip install intrinio_sdk
!pip install backtrader
!pip install cufflinks
```

```
Requirement already satisfied: configparser in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (3.7.4)
WARNING: You are using pip version 19.3.1; however, version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Requirement already satisfied: intrinio_sdk in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (5.5.0)
Requirement already satisfied: python-dateutil in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from intrinio_
sdk) (2.8.1)
Requirement already satisfied: six>=1.10 in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from intrinio_sdk)
 (1.11.0)
Requirement already satisfied: urllib3>=1.15 in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from intrinio_sd
k) (1.23)
Requirement already satisfied: certifi in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from intrinio_sdk) (20
18.10.15)
WARNING: You are using pip version 19.3.1; however, version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Requirement already satisfied: backtrader in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (1.9.74.123)
WARNING: You are using pip version 19.3.1; however, version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Requirement already satisfied: cufflinks in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (0.17.3)
Requirement already satisfied: ipython>=5.3.0 in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from cufflinks)
```

In [5]:
```
import intrinio_sdk
import configparser as cp
import statsmodels.api as sm
import numpy as np
import pandas as pd
import cufflinks as cf
import matplotlib.pyplot as plt
cf.set_config_file(offline=True)
```

In [6]:
```
cfg = cp.ConfigParser()
cfg.read('../resources/credentials.cfg')
```

Out[6]:
```
['../resources/credentials.cfg']
```

3.1. Use Intrinio API to pull the daily stock prices of all the constituents of the selected industry

In [7]:
```
API_KEY = cfg['intrinio']['app_key']
intrinio_sdk.ApiClient().configuration.api_key['api_key'] = API_KEY
security_api = intrinio_sdk.SecurityApi()
```

In [8]:
```
sp_df= pd.read_csv("../data/SP1500.csv")
data = sp_df[sp_df['industry']=='Data Processing and Outsourced Services']
```

In [9]:
```
tickers= list(data['ticker'])[1:]
tickers
```

Out[9]:
```
['ADP',
 'BR',
 'CATM',
 'CSGS',
 'EVTC',
 'EXLS',
 'FIS',
 'FISV',
 'FLT',
 'GPN',
 'JKHY',
 'MA',
 'MMS',
 'EGOV',
 'PAYX',
 'PYPL',
 'SABR',
 'SYKE',
 'WU',
 'TTEC',
 'V',
 'WEX']
```

In [10]:
```
# date | Return prices on or after the date (optional)
start_date = '2019-11-15'

# date | Return prices on or before the date (optional)
end_date = '2020-04-30'

# str | Return stock prices in the given frequency (optional) (default to daily)
frequency = 'daily'
```

In [11]:
```
dfs = []

for ticker in tickers:
    next_page = ''
    response = security_api.get_security_stock_prices(ticker,
                                                      start_date = start_date,
                                                      end_date = end_date,
                                                      )
    df = [p.to_dict() for p in response.stock_prices]
    next_page = response.next_page
    if next_page != None:
        response = security_api.get_security_stock_prices(ticker,
                                                          start_date = start_date,
                                                          end_date = end_date,
                                                          next_page = next_page,
                                                          )
        df.extend(p.to_dict() for p in response.stock_prices)
    df = pd.DataFrame.from_dict(df)
    df['secid'] = ticker
    dfs.append(df)
```

In [12]:
```
data_df = pd.concat(dfs)
data_df.index = pd.DatetimeIndex(data_df['date'])
data_df = data_df.drop('date', axis=1)
data_df.index.name = None
data_df = data_df.sort_index()
data_df.shape
```

Out[12]:
```
(2508, 13)
```

In [13]:
```
# Check: number of constituents in the Healthcare Services industry (More than 10)
len(data_df['secid'].unique())
```

Out[13]:
```
22
```

In [14]:
```
# Make sure that there are at least 100 trading days.
len(data_df.index)/len(data_df['secid'].unique())
```

Out[14]:
```
114.0
```

3.2 The long/short Trading Strategy for the Select company (stock) from Data Processing and Outsourced Services industry.

The selected company: Visa Inc.(Ticker: V) and reasons for this choice.

The selected company is Visa Inc.(Ticker: V). This company would be a good choose to analyze as based on the analysis conducted in Part 2, this company has the largest Market capitalization and a good P/E ratio among the whole industry, which is a representative company for this industry. Additionally, based on analysis of its stock price trend, the stock would go up in the future and bring out expected returns.Therefore, this is a good stock

to conduct trading strategy.

In [15]:
```
v_df = data_df[data_df['secid'] == "V"]
data_df.shape
```

Out[15]: (2508, 13)

In [16]:
```
v_df.head()
```

Out[16]:

	adj_close	adj_high	adj_low	adj_open	adj_volume	close	frequency	high	intraperiod	low	open	volume	secid
2019-11-15	179.510016	180.418700	178.821014	179.769640	7809545.0	179.77	daily	180.6800	False	179.0800	180.03	7809545.0	V
2019-11-18	179.400175	180.062416	178.721159	179.310306	7175036.0	179.66	daily	180.3232	False	178.9800	179.57	7175036.0	V
2019-11-19	182.505678	182.715374	179.941691	180.129120	8551440.0	182.77	daily	182.9800	False	180.2023	180.39	8551440.0	V
2019-11-20	181.397283	182.955027	180.089178	182.046343	6040751.0	181.66	daily	183.2200	False	180.3500	182.31	6040751.0	V
2019-11-21	179.629843	181.547066	179.055674	180.907992	5112247.0	179.89	daily	181.8100	False	179.3150	181.17	5112247.0	V

Strategy and signals for a 2 day short and 5 long Portfolio

In [17]:
```
short_window = 2
long_window = 5
```

In [18]:
```
signals = pd.DataFrame(index = v_df.index)
signals['signal'] = 0.0
```

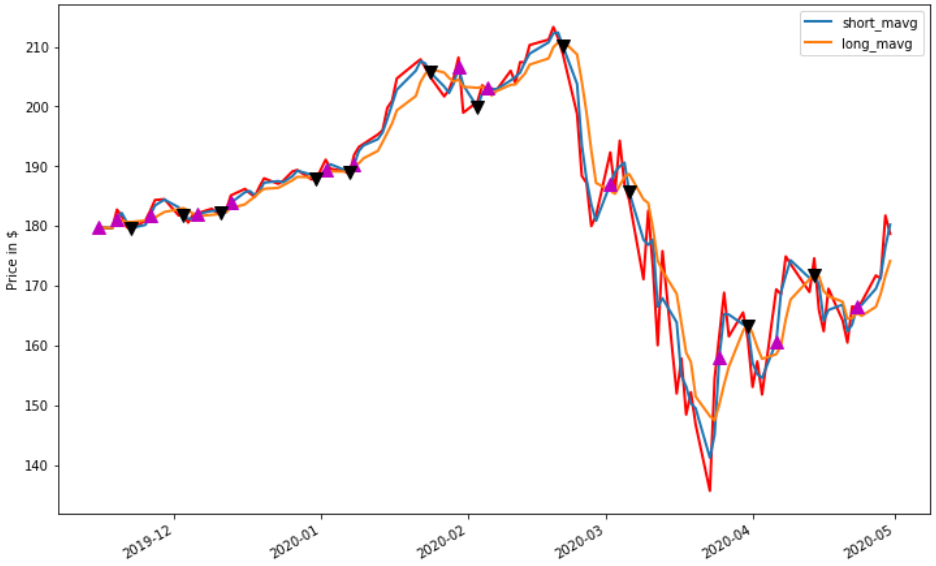
In [19]:
```
signals['short_mavg'] = v_df['close'].rolling(window = short_window,
                                              min_periods = 1,
                                              center = False).mean()
```

In [20]:
```
signals['long_mavg'] = v_df['close'].rolling(window = long_window,
                                             min_periods = 1,
                                             center = False).mean()
```

In [21]:
```
signals['signal'][short_window:] = np.where(signals['short_mavg'][short_window:]
                                            > signals['long_mavg'][short_window:],
                                            1.0, 0.0)
```

In [22]:
```
signals['positions'] = signals['signal'].diff()
signals['positions'][0] = 1
```

In [23]:
```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(111,  ylabel='Price in $')
v_df['close'].plot(ax=ax1, color='r', lw=2.)
signals[['short_mavg', 'long_mavg']].plot(ax=ax1, lw=2.)
ax1.plot(signals.loc[signals.positions == 1.0].index,
         signals.short_mavg[signals.positions == 1.0],
         '^', markersize=10, color='m')
ax1.plot(signals.loc[signals.positions == -1.0].index,
         signals.short_mavg[signals.positions == -1.0],
         'v', markersize=10, color='k')
plt.show()
plt.savefig("../graph/strategy_signal_Visa.jpg")
```




```
<Figure size 432x288 with 0 Axes>
```

3.3 Backtest a 100,000 long/short portfolio using Pandas and plot the portfolio value.

In [24]:
```
initial_capital= float(100000.0)
shares = 100
```

In [25]:
```
positions = pd.DataFrame(index = signals.index).fillna(0.0)
positions['V'] = shares * signals['signal']
```

In [26]:
```
portfolio = pd.DataFrame(index = positions.index).fillna(0.0)
pos_diff = positions.diff()
portfolio['holdings'] = positions.multiply(v_df['adj_close'], axis=0)
portfolio['cash'] = initial_capital - (pos_diff.multiply(v_df['adj_close'],
                                                         axis=0)).sum(axis=1).cumsum()
portfolio['total'] = portfolio['cash'] + portfolio['holdings']
portfolio['returns'] = portfolio['total'].pct_change()
portfolio['signal'] = [ 'short' if i == 0 else 'long' for i in signals['signal']]
```

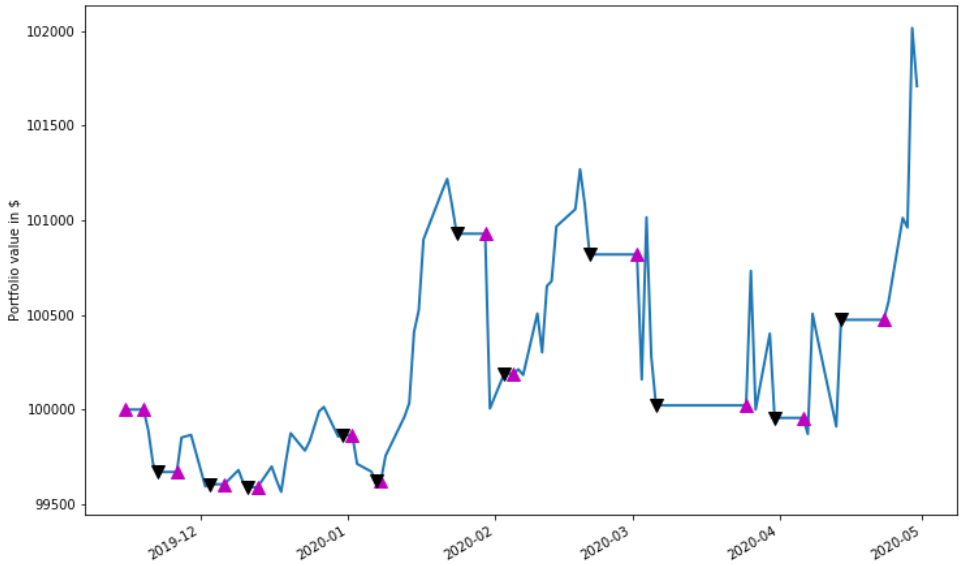
In [27]:
```
portfolio.head()
```

Out[27]:

	holdings	cash	total	returns	signal
2019-11-15	0.000000	100000.000000	100000.000000	NaN	short
2019-11-18	0.000000	100000.000000	100000.000000	0.000000	short
2019-11-19	18250.567779	81749.432221	100000.000000	0.000000	long
2019-11-20	18139.728307	81749.432221	99889.160528	-0.001108	long
2019-11-21	17962.984285	81749.432221	99712.416506	-0.001769	long

In [28]:
```
fig = plt.figure(figsize=(12,8))

ax1 = fig.add_subplot(111, ylabel='Portfolio value in $')
portfolio['total'].plot(ax=ax1, lw=2.)
ax1.plot(portfolio.loc[signals.positions == 1.0].index,
         portfolio.total[signals.positions == 1.0],
         '^', markersize=10, color='m')
ax1.plot(portfolio.loc[signals.positions == -1.0].index,
         portfolio.total[signals.positions == -1.0],
         'v', markersize=10, color='k')
plt.show()
plt.savefig("../graph/backtest_Visa.jpg")
```




```
<Figure size 432x288 with 0 Axes>
```

3.4 Evaluate your trading strategy, calculate the portfolio sharpe ratio, maximum drawdown and Compound Annual Growth Rate (CAGR). In a short paragraph interpret the results.

In [29]:
```
# Isolate the returns of your strategy
returns = portfolio['returns']

# 100 Days Sharpe ratio
sharpe_ratio = np.sqrt(127) * (returns.mean() / returns.std())
print('120 days sharpe ratio: {}'.format(sharpe_ratio))

# annualized Sharpe ratio
sharpe_ratio = np.sqrt(252) * (returns.mean() / returns.std())
print('annualized sharpe ratio: {}'.format(sharpe_ratio))
```

```
120 days sharpe ratio: 0.6623641645924543
annualized sharpe ratio: 0.9330292051666959
```

For 127 Trading days - Sharpe Ratio of 0.6623641645924543

The sharp ratio of the portfolio in 120 days is a litte below the standard range: 1.25-1.75. This shows the portfolio should lower its volatility per unit for the excess of risk-free return.

For 252 Trading days - Sharpe Ratio of 0.9330292051666959

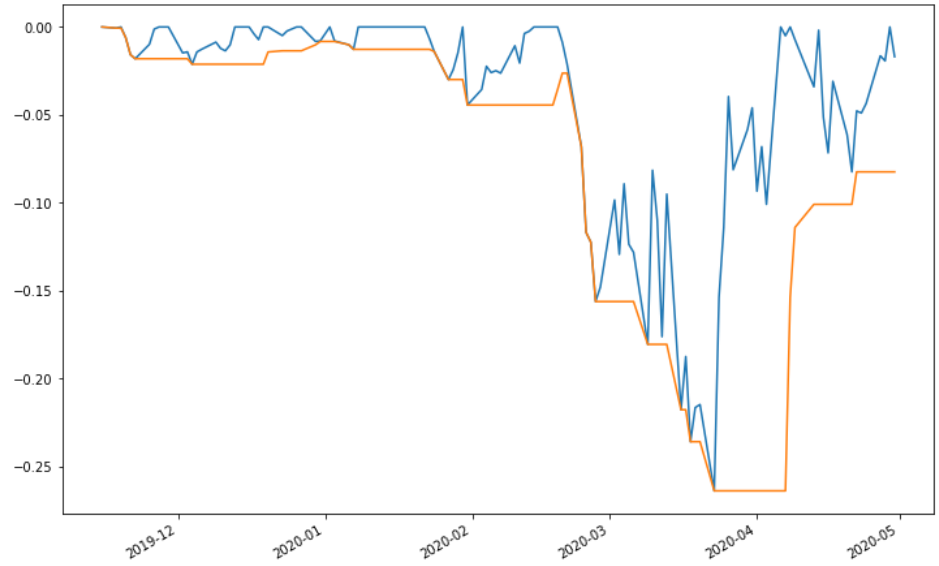
This portfolio final value shows that competent at investing with all of the above techniques, plus futures, mortgage-backed securities, asset-backed securities, interest rate swaps, credit default swaps, and short selling of cash and debt instruments (via the repurchase market) to finance long positions.

Maximum Drawdown¶

In [30]:
```
max_window = 12

rolling_max = v_df['adj_close'].rolling(max_window, min_periods=1).max()
daily_drawdown = v_df['adj_close']/rolling_max - 1.0

max_daily_drawdown = daily_drawdown.rolling(max_window, min_periods=1).min()
fig = plt.figure(figsize=(12,8))
daily_drawdown.plot()
max_daily_drawdown.plot()
plt.show()
plt.savefig("../graph/maximum_drawdown_Visa.jpg")
```




```
<Figure size 432x288 with 0 Axes>
```

Maximum Drawdown:

From the chart we can see, the daily maximum drawdown is about 35%, which shows the maximum of the potential loss of the investment would be 35% of the whole portfolio and this is kind of high risk investment.**

Compound Annual Growth Rate (CAGR)

In [31]:
```
days = (v_df.index[-1] - v_df.index[0]).days

cagr = ((((v_df['adj_close'][-1]) / v_df['adj_close'][1])) ** (365.0/days)) - 1
print(cagr)
```

```
-0.008267943734828309
```

Compound Annual Growth Rate (CAGR)

The CAGR is negative, which means this stratgy doesn't perform very well for this industry and using this strategy. Based on realistic reason, for real investment, the selection of industries would significantly affect the return performance.

3.5 Use backtrader to backtest a portfolio based on the industry that you selected. Make sure that there are executed trades and upload the plot that backtrader generated (use provided notebook)

Strategy Buy when 2 day moving average greater than 5 day moving average

Sell when 5 day moving average greater than 2 day moving average

Trade Management and Position Sizing

- Position size buy/sell 100 shares at a time without
- Backtrader will reject order if not enough funds

In [32]:
```
import backtrader as bt
import pandas as pd
```

In [33]:
```
data_df = pd.read_csv("../data/data_df.csv", index_col= 0)
```

In [34]:
```
data_df.columns
```

Out[34]:
```
Index(['adj_close', 'adj_high', 'adj_low', 'adj_open', 'adj_volume', 'close',
       'frequency', 'high', 'intraperiod', 'low', 'open', 'volume', 'secid'],
      dtype='object')
```

In [35]:
```
data_df['secid'].unique()
```

Out[35]:
```
array(['WEX', 'SYKE', 'CATM', 'ADS', 'V', 'GPN', 'JKHY', 'EVTC', 'PYPL',
       'MA', 'FLT', 'EXLS', 'MMS', 'SABR', 'CSGS', 'FISV', 'PAYX', 'FIS',
       'ADP', 'BR', 'TTEC', 'WU', 'EGOV'], dtype=object)
```

In [36]:
```
bt_data = data_df[['adj_open', 'adj_high', 'adj_low', 'adj_close', 'adj_volume', 'secid']]

bt_data = bt_data.rename(columns={'adj_open':'open', 'adj_high':'high', 'adj_low':'low',
                                  'adj_close':'close', 'adj_volume':'volume', 'secid':'name'})
bt_data.index.name = None
bt_data.to_csv("../data/bt_data.csv")
```

In [37]:
```
bt_data = pd.read_csv("../data/bt_data.csv", index_col = 0)
bt_data.index = pd.DatetimeIndex(bt_data.index)
bt_data.head()
```

Out[37]:

	open	high	low	close	volume	name
2019-11-15	197.420000	200.830000	195.950000	200.550000	269874.0	WEX
2019-11-15	35.740000	35.740000	35.070000	35.170000	705534.0	SYKE
2019-11-15	40.190000	40.750000	39.940000	40.600000	362497.0	CATM
2019-11-15	106.223361	106.223361	102.217745	102.824054	1486284.0	ADS
2019-11-15	179.769640	180.418700	178.821014	179.510016	7809545.0	V

In [38]:
```
class SMAStrategy(bt.Strategy):

    def __init__(self):
        signal_short = bt.ind.SMA(period = 5)
        signal_long = bt.ind.SMA(period = 8)
        self.sma_diff = signal_short - signal_long

    def next(self):
        if not self.position:
            if self.sma_diff >= 0:
                self.buy(size = 100)
        else:
            if self.sma_diff < 0:
                self.sell(size = 100)
```

In [39]:
```
constituents = bt_data['name'].unique()
for stock in constituents:
     bt_data[bt_data['name'] == stock].to_csv("../{}_bt.csv".format(stock))
```

In [52]:
```
constituents = bt_data['name'].unique()
for stock in constituents:
     bt_data[bt_data['name'] == stock].to_csv("../{}_bt.csv".format(stock))
```

In [53]:
```
bt_data = pd.read_csv("../data/bt_data.csv",index_col = 0 )
constituents = bt_data.name.unique()
len(constituents)
```

Out[53]:
```
23
```

In [42]:
```
class MomentumStrategy(bt.Strategy):
    params = dict(
        num_universe = 5, # Number of Industry Constituents
        num_positions = 1, # Set the number of position to hold at any given time
        when = bt.timer.SESSION_START,
        weekdays = [5],
        weekcarry = True,
        rsi_period = 8, # Relative Strength Index Periods
        sma_period = 18 # Moving Average Periods
    )

    def __init__(self):
        self.inds = {}
        self.rsi = {}

        self.securities = self.datas[1:]
        for s in self.securities:
            self.inds[s] = {}
            self.inds[s]['sma'] = bt.ind.SMA(s, period = self.p.sma_period)
            self.inds[s]['sma'].plotinfo.plot = False
            self.inds[s]['rsi'] = bt.ind.RSI(s, period = self.p.rsi_period)
            self.inds[s]['rsi'].plotinfo.plot = False

        self.add_timer(
            when = self.p.when,
            weekdays = self.p.weekdays,
            weekcarry = self.p.weekcarry
        )

    def notify_timer(self, timer, when, *args, **kwargs):
            self.rebalance()

    def notify_trade(self, trade):
        if trade.size == 0:
            print("DATE:", trade.data.datetime.date(ago=0),
                  " TICKER:", trade.data.p.name,
                  "\tPROFIT:", trade.pnlcomm)

    def rebalance(self):

        rankings = list(self.securities)

        rankings.sort(
            key = lambda s: self.inds[s]['sma'][0],
            reverse = False
        )

        rankings = rankings[:self.p.num_universe]

        rankings.sort(
            key = lambda s: self.inds[s]['rsi'][0],
            reverse = True
        )

        # position size short
        pos_size = -1 / self.p.num_positions

        # Sell when ranking
        for i, d in enumerate(rankings):
            if self.getposition(d).size:
                if i > self.p.num_positions:
                    self.close(d)

        # Buy and rebalance stocks with remaining cash
        for i, d in enumerate(rankings[:self.p.num_positions]):
            self.order_target_percent(d, target = pos_size)
```

In [43]:
```
starcash = 100000
```

In [44]:
```
cerebro = bt.Cerebro()
cerebro.broker.setcash(starcash)
cerebro.broker.setcommission(commission=0.0)
```

In [45]:
```
first_stock = True
for stock in constituents:
    # Load the each stock price data from the btdata folder
    filename = "../{}_bt.csv".format(stock)
    data = bt.feeds.GenericCSVData(
        dataname = filename,
        dtformat = ('%Y-%m-%d'),
        datetime = 0,
        high = 2,
        low = 3,
        open = 1,
        close = 4,
        volume = 5,
        openinterest = -1,
        name = stock)
    if first_stock:
        data0 = data
        data.plotinfo.sameaxis = False
        data.plotinfo.plotylimited = True
        first_stock = False
    else:
        data.plotinfo.plotmaster = data0
        data.plotinfo.subplot = False
        data.plotinfo.sameaxis = False
        data.plotinfo.plotylimited = True
    cerebro.adddata(data, name = stock)
```

In [46]:
```
cerebro.addstrategy(MomentumStrategy)
```

Out[46]:
```
0
```

In [47]:
```
cerebro.addanalyzer(bt.analyzers.SharpeRatio, riskfreerate=0.0)
cerebro.addanalyzer(bt.analyzers.Returns)
cerebro.addanalyzer(bt.analyzers.DrawDown)
```

In [48]:
```
print('Starting Portfolio Value: %.2f' % cerebro.broker.getvalue())
result = cerebro.run()
print('Ending Portfolio Value: %.2f' % cerebro.broker.getvalue())
```

```
Starting Portfolio Value: 100000.00
DATE: 2019-12-02  TICKER: SYKE  PROFIT: -1425.1199999999908
DATE: 2019-12-09  TICKER: CATM  PROFIT: 3651.9599999999887
DATE: 2019-12-16  TICKER: SABR  PROFIT: -221.8994837875405
DATE: 2019-12-30  TICKER: SYKE  PROFIT: 409.49999999999613
DATE: 2020-01-21  TICKER: EVTC  PROFIT: -2187.835053436096
DATE: 2020-02-10  TICKER: EVTC  PROFIT: -1164.416194261794
DATE: 2020-02-18  TICKER: WU    PROFIT: 3924.2021742226625
DATE: 2020-03-09  TICKER: EVTC  PROFIT: 11353.090177659516
DATE: 2020-03-23  TICKER: SYKE  PROFIT: 32181.840000000022
DATE: 2020-04-06  TICKER: WU    PROFIT: 11339.959999999992
Ending Portfolio Value: 125407.79
```

In [49]:
```
dd = result[0].analyzers.drawdown.get_analysis()['max']['drawdown']
cagr = result[0].analyzers.returns.get_analysis()['rnorm100']
sharpe = result[0].analyzers.sharperatio.get_analysis()['sharperatio']
print("Max Drawdown: {}%\nCAGR: {}%\nSharpe: {}".format(dd, cagr, sharpe))
```

```
Max Drawdown: 32.77521085652766%
CAGR: 64.94870792734962%
Sharpe: 0.8482416072795671
```

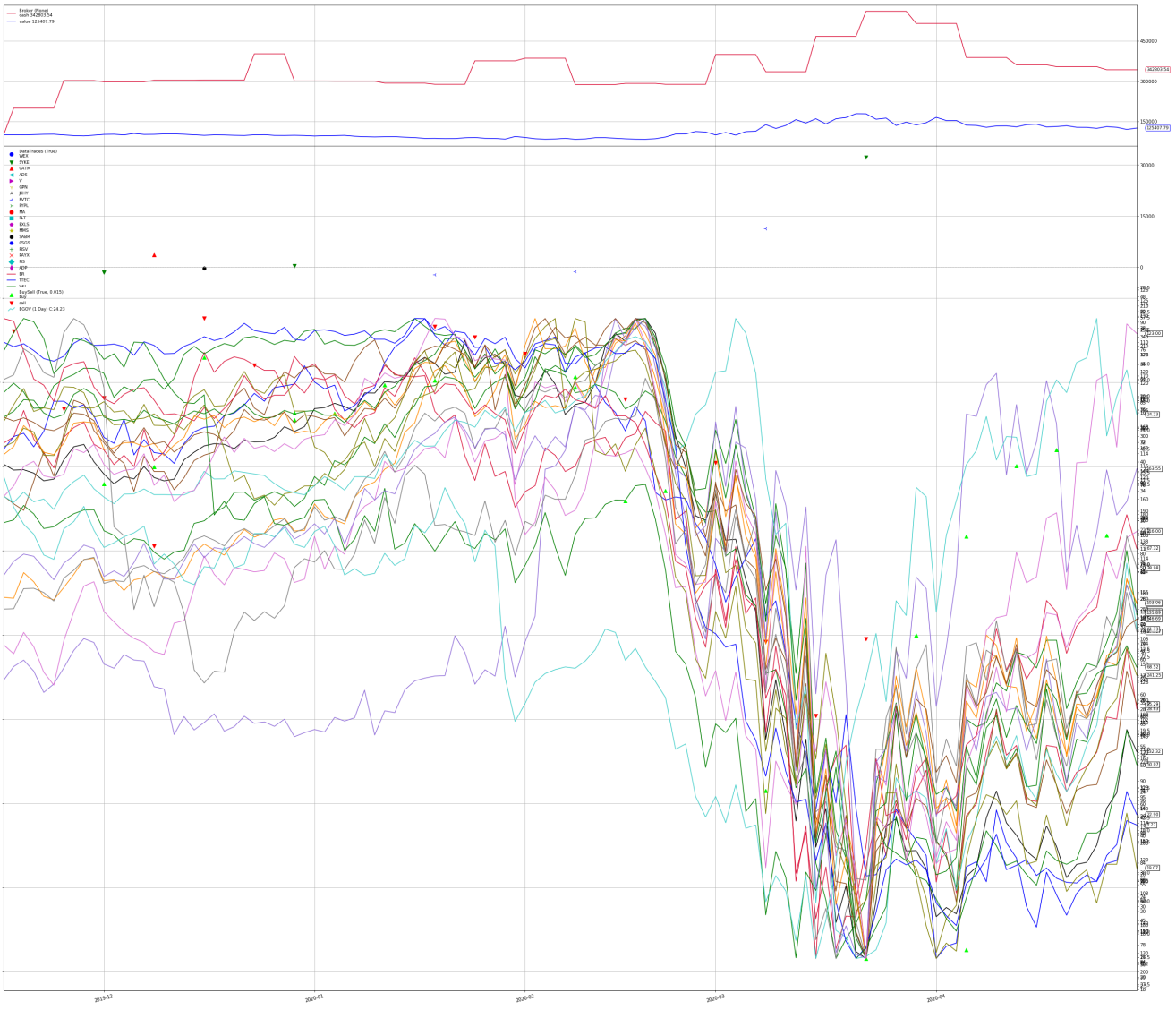
In [50]:
```
plt.rcParams['figure.figsize'] = [38, 32]
cerebro.plot(volume=False)
plt.savefig("../graph/backtrader_industry.jpg")
```

```
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/backtrader/plot/__init__.py:30: UserWarning:

matplotlib.pyplot as already been imported, this call will have no effect.

/home/nbuser/anaconda3_501/lib/python3.6/site-packages/backtrader/plot/plot.py:127: UserWarning:

matplotlib.pyplot as already been imported, this call will have no effect.
```



```
<Figure size 2736x2304 with 0 Axes>
```

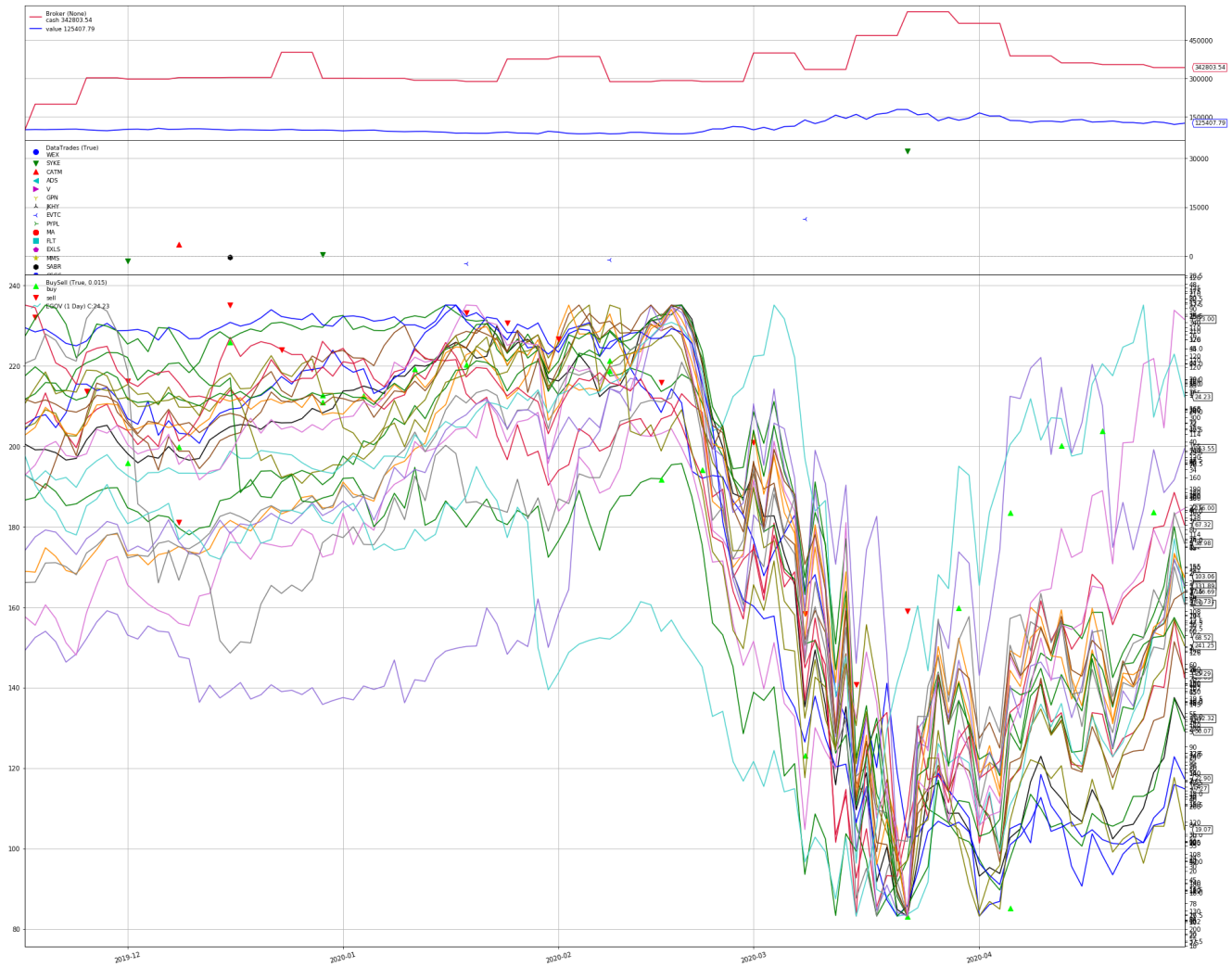
In [51]:
```
dd = result[0].analyzers.drawdown.get_analysis()['max']['drawdown']
cagr = result[0].analyzers.returns.get_analysis()['rnorm100']
sharpe = result[0].analyzers.sharperatio.get_analysis()['sharperatio']

print("Max Drawdown: =dd={}, carg={}, sharpe{}".format(dd, cagr, sharpe))
plt.rcParams['figure.figsize'] = [28, 22]
cerebro.plot(volume=False)
plt.savefig("../graph/backtrader_industry_ratios.jpg")
```

```
Max Drawdown: =dd=32.77521085652766, carg=64.94870792734962, sharpe0.8482416072795671

/home/nbuser/anaconda3_501/lib/python3.6/site-packages/backtrader/plot/plot.py:127: UserWarning:

matplotlib.pyplot as already been imported, this call will have no effect.
```




```
<Figure size 2016x1584 with 0 Axes>
```

In [ ]:
```

```